Brandenburg
University of Technology
Cottbus - Senftenberg

Dealing with Time in the Multiple Traveling Salesmen Problem with Moving Targets

Anke Stieber
Armin Fügenschuh

Cottbus Mathematical Preprints
COMP # 4(2019)

Dealing with Time in the Multiple Traveling Salesmen Problem with Moving Targets

Anke Stieber* Armin Fügenschuh†

April 18, 2019

Abstract

The multiple traveling salesmen problem with moving targets is a generalization of the classical traveling salesmen problem, where the targets (cities or objects) are moving over time. Additionally, for each target a visibility time window is given. The task is to find routes for several salesmen so that each target is reached exactly once within its visibility time window and the sum of all traveled distances of all salesmen is minimal. We present different modeling formulations for this TSP variant. The time requirements are modeled differently in each approach. Our goal is to examine what formulation is most suitable in terms of runtime to solve the multiple traveling salesmen problem with moving targets with exact methods. Computational experiments are carried out on randomly generated test instances to compare the different modeling approaches. The results for large-scale instances show, that the best way to model time requirements is to directly insert them into a formulation with discrete time steps.

Keywords: Dynamic traveling salesmen problem, moving targets, time-relaxation, integer linear programming, second-order cone programming.

1 Introduction

In the classical TSP the targets (cities, points) are static entities and only one salesman is considered to traverse each target exactly once. The TSP is an ordering problem and the time aspect is not considered in any way. For a survey on the TSP we refer to Lawler et al. [22] or Reinelt [28].

This research deals with a dynamic variant of the TSP, where time plays an important role. First of all, the targets are not fixed, they move continuously on arbitrarily shaped trajectories with a certain variable speed function. Secondly each target is assigned a visibility time window. That means a salesman can only intercept the target within its respective time interval. During visibility targets move on trajectories. A time window can also be interpreted in a way that a target enters a system (or a certain space) and later on is removed from the system. Additionally, we are dealing with the case of multiple salesmen. This TSP generalization is called multiple traveling salesmen problem with moving targets (MTSPMT) and time windows.

Each salesman starts his tour from an initial depot, but does not have to finish at the depot. Each target must be visited once by exactly one salesman within its visibility window. The objective is to minimize the total traveled distances of all salesmen. Obviously, if we restrict the number of salesman to one, the position of all cities to be fixed over time and extend all visibility windows to the whole time horizon, we obtain the classical TSP, which is NP-hard, see Garey and Johnson [12]. Thus, the MTSPMT as a generalization of the classical TSP is NP-hard, too.

The MTSPMT as a dynamic generalization of the TSP is suitable for real-world problems. A possible application of the MTSPMT can be found in the defense sector. An area, e.g., an airport or a military base, must be protected from incoming hostile rocket, artillery, or mortar (RAM) fire. With a battery of laser guns deployed within or nearby the protected area the decision has to be made, which available laser to select for the countermeasure to protect the area. Here, laser guns correspond to the salesmen and the incoming fire entities are the moving targets having a certain

*A. Stieber, Helmut Schmidt University/University of the Federal Armed Forces, Hamburg, Germany, anke.stieber@hsu-hh.de

†A. Fügenschuh, Brandenburg University of Technology Cottbus-Senftenberg, Cottbus, Germany, fuegenschuh@b-tu.de

visibility time window. The time window starts at that moment at which reachability by the laser guns is guaranteed and ends at the latest point in time where a destruction of the target is possible (before impact). Each target moves on a certain trajectory assuming a previous radar-detection. From a mathematical point of view, this application is an online optimization problem, where the complete data of the problem instance is not given in advance and a decision has to be made immediately. It can be solved "offline" and new data is then integrated into the offline algorithm at runtime by a moving horizon approach. For a detailed description of the application we refer to Stieber et al. [31].

Helvig et al. [15] addressed the Moving-Target TSP, which is the MTSPMT restricted to one salesman. As possible applications they mention a supply ship, that resupplies patrolling boats or an airplane that must intercept a number of mobile ground units. They also addressed the Multi-Pursuer Moving-Target TSP with Resupply, where multiple pursuers are considered and each pursuer must return to the origin for resupply after intercepting a target.

Many practical applications such as routing and scheduling of vehicles have a time-dynamic component inherent. In the last two decades logistics distributions were faced with an increasing traffic load, traffic congestion and uncertainty in travel times caused by bad weather conditions or other random incidences. The multiple Vehicle Routing Problem (VRP) with time-dependent traveling times and time windows is very similar to the MTSPMT. However, the movement of a target does not only influence the length of a certain arc and thus the travel time to a certain target, but to all other targets as well. Since all targets are moving simultaneously and constantly the length of an arc has two degrees of freedom, i.e., the length of an arc is determined by the time the arc is entered and the time the arc is left. Both these times exactly correspond to two spatial positions of the incident targets. Thus, for the MTSPMT we have varying distances between two targets and varying travel times and both values do not correspond since waiting is included. Assuming a salesman located at a certain position is traveling towards a target, which is going to be very close to the salesman position in a little while. Being on the way to that target, it is advantageous for the salesman to slow down to catch the target when it is nearby than to speed up and travel a long distance. On the other hand, when the target is heading away from the salesman it is better to use maximum speed and catch the target as early as possible. As for the multiple VRP with time-dependent traveling time and time windows, traveling times change for each arc individually over time, however, for the MTSPMT the distances change over time as well. For the MTSPMT minimizing the travel time is different from minimizing the distance traveled.

In this research paper we investigate the time aspect of the MTSPMT in modeling when minimizing the traveled distance. Based on different ways of integrating time into the model formulation, four different variants are presented and compared regarding performance. Two model approaches have already been published by Stieber et al. [31], the time-discrete (TD) model and the time-continuous (TC) model. In addition, we generate time-free models. For these models we relax the time requirements completely and use subprograms to check and create time feasibility. These feasibility checker are based on discrete time steps on one hand and on the other hand we use a continuous time formulation. The two resulting modeling variants are called time-free model with time-discrete feasibility checking (TFTD) and time-free model with time-continuous feasibility checking (TFTC). The contributions of this research paper are in detail:

1. The TC model from Stieber et al. [31] is tightened, such that the distance coefficients are only dependent on the arrival times of the salesmen (departure time is equal to the former arrival time).

2. Furthermore, we reformulate the TC model as a second-order cone program (SOCP), which can be solved by the standard mixed-integer linear programming (MILP) solver IBM ILOG CPLEX. This procedure requires some assumptions of the target movements.

3. The time-free modeling approach is proposed, which is a two-stage variant. In the first stage time is completely relaxed. A time-feasible solution is computed in a second stage by subproblems integrating not all but necessary time restrictions. In case of a discrete feasibility checker we obtain the TFTD model and in the continuous case we have the TFTC model, respectively.

4. The optimal solutions of the two-stage models are computed using a branch-and-bound framework. We present an algorithm, that makes use of a callback function (an advanced feature of CPLEX).

5. Computational experiments are carried out using randomly generated problem instances with a varying number of targets, salesmen and time steps.

The aim of this research is to find a formulation to solve MTSPMT instances with the best performance in terms of CPU time. The remainder of this article is organized as follows. In Section 2 we provide a survey of the relevant literature. The two model variants TD and TC, which incorporate the time as a discrete and a continuous concept are recalled and extended in Section 3. The time-free approach (TFTD and TFTC) and its solution method are addressed in Section 4 and 5. The computational experiments are presented in Section 6 and we draw conclusions in Section 7.

2 Related Work

This research article addresses a generalization of the classical traveling salesman problem (TSP) by considering multiple salesmen and moving targets with time windows. For a survey on the classical TSP we refer to Reinelt [28].

There are a number of articles in the literature that deal with dynamical TSP generalizations. However, the TSP literature concerning moving targets is less developed [16, 18, 8]. The first articles addressing the traveling salesman problem with moving targets (TSPMT) are Helvig et al. [15, 16]. The authors present an exact and an approximation algorithm for the one-dimensional case, where targets and salesmen move on a straight line. Other specific variants of the TSPMT with restrictions for target speed, movement and the number of targets are also addressed, e.g., the targets move towards the origin (starting point of the salesman), never reach the origin, or the TSPMT with resupply, where the persuer must return to the origin after intercepting a target. The proposed algorithms are not applicable to the general case of the TSPMT. Specific variants of the TSPMT and the TSPMT with resupply are also addressed by Jiang et al. [18], Jindal et al. [19] and Englot et al. [8]. Solution approaches are mainly heuristics such as genetic algorithms.

Some articles address a stochastic variant of the dynamic traveling salesman problem. Here, locations of the targets change over time caused by stochastic processes. For example, problem instances in Ahrens [2] were generated from static TSP datasets originated from a published and standardized library. The movement of each target is modeled by a Gaussian-distributed random distance vector that is added to its location. Time is integrated in a way that the movement from one target to another can be done in one time step and the targets localized in the 2-dimensional space move at each time step. The instances were solved by heuristics in an online calculation. The author examined the applicability of standard (static) TSP solvers to the dynamic instances. Computational experiments were carried out with the Tour Construction Framework which combines global and local heuristics and the TSP tour construction heuristic "nearest neighbor".

In the case of describing the MTSPMT with discrete times steps, we obtain a copy of each target for each time step in the respective time window. Let us consider the copies of each target as a subset (or cluster). To this end, we have a set of clusters and the task is still that each cluster has to be visited once by exactly one salesman. In case of considering one salesman, this gives us another TSP variant, namely the equality generalized TSP (E-GTSP). Since we consider discrete time steps, we also have discrete trajectory positions of the targets. Any such position is characterized by the target number and the time step. Considering two positions of different targets as two nodes then there is an arc between these nodes, if and only if a salesman is able to travel the euklidean distance of the nodes within the time difference. That means, the salesman is not allowed to exceed its maximum speed value. Generally, there is no anti-parallel arc due to the progression of time. In this context an instance of the TSPMT (MTSPMT with one salesman) can be formulated as an instance of the asymmetrical E-GTSP. With the restriction that at least one element per subset has to be visited we obtain the generalized TSP (GTSP), this problem is also known as set TSP or group TSP. The asymmetrical GTSP and E-GTSP have been investigated in Laporte et al. [21]. Noon and Bean [26] showed, that any problem that can be modeled as a GTSP, can be transformed into an asymmetrical TSP. The equivalent problem with more than one salesman is the generalized vehicle routing problem (GVRP), see Ghiani and Improta [13]. For the symmetrical case there is a very recent contribution by Sundar and Rathinam [32]. The authors addressed the generalized multiple depot TSP (GMDTSP) and provide a polyhedral study for this problem class. Presenting a branch-and-bound approach that was also realized by the callback functionality of CPLEX, computational experiments were carried out for 14 to 105 targets (which

correspond to the number of target copies in our notation) and a maximum number of 21 clusters (which correspond to the number of targets in our notation).

In Picard and Queyranne [27] a similar problem is considered: the time dependent traveling salesman problem (TDTSP). Here, a complete graph is considered and the cost values (or the travel times) depend on the position of the target in the tour. Thus, there are number of targets many discrete time steps and only a discrete number of cost values. The authors use shortest paths in a multipartite network for the solution with branch-and-bound and relaxation methods. Abeledo et al. [1] is based on the formulation given by Picard and Queyranne [27] and provides a study of the TDTSP polytope. The authors present several facet defining inequalities and perform computational experiments with their branch-and-cut-and-price algorithm.

A generalization of the TDTSP is the time dependent vehicle routing problem (TDVRP). Here, more than one salesman (vehicle) is considered and capacity restrictions as well as time windows are imposed. Malandraki and Daskin first formulated the TDVRP [23] as a mixed-integer linear program. They used step functions to model the travel times depending on the distance between the nodes and the time of the day. Furthermore, a nearest-neighbor heuristic was presented. Due to the computational complexity of TDVRP scientific contributions mainly focus on heuristic approaches. Some exemplary articles follow.

Ichoua et al. [17] integrated the time dependency in the travel speed instead of in the travel time. The travel speeds were modeled by step-functions of the time of the day, leading to piece-wise linear functions of the travel times. They report on a tabu search algorithm, that was adapted to the time-dependent model and experiments were conducted in a static and dynamic environment.

With regard to better reliability in scheduling and routing problems, Fleischmann et al. [9] used time-varying continuous travel times and retrieved the information from a traffic information system that was tested in the city of Berlin. Computational results were reported for several VRP heuristics.

Jung and Haghani [20, 14] also applied continuous travel time functions and gave a MILP formulation of the TDVRP with discrete time steps. A genetic algorithm was presented and the results were compared with an exact solution method for small and mid-sized instances and for bigger instances a lower-bound procedure was used. Instances with 5 to 30 nodes and 10, 15 and 30 times steps were considered.

A recent article deals with a more realistic modeling of the time-dependent travel times. Mancini [24] models the travel times by a polynomial instead of linear functions, computational results were carried out by a heuristic method.

Very few literature is published concerning exact methods, some of them are e.g., Albiach et al. [3] and Soler et al. [29]. Based on the work of Noon and Bean [26] they provide a theoretical work of transforming an instance of the asymmetric TDTSP with time windows or the TDVRP with time windows into an instance of the Asymmetric TSP (ATSP) or VRP (AVRP) respectively. Soler et al. [29] is a generalization of the first one, because it deals with multiple salesmen. The conversions are carried out by transforming the underlying graph into an instance of the GTSP or the GVRP respectively, and then into the ATSP and the AVRP. Albiach et al. [3] also performed computational experiments using an exact algorithm for the Mixed General Routing Problem. Instances with up to 222 vertices and one salesman were considered, where the number of arcs is between 5 and 20% of the arcs of a complete graph.

Van Woensel et al. [33] introduced a new approach to model potential traffic congestion. This model is based on a queueing approach to traffic flows. Computational experiments were carried out for small instances with 10 cities using explicit enumeration and for up to 100 cities with an ant colony heuristic.

In summary there are only few research contributions to the MTSPMT. Articles on the MTSPMT or the TSPMT mainly consider specific problems or impose restrictions on the movement of the targets. The MTSPMT in general is closely related to the time dependent TSP and VRP; the MTSPMT with discrete time steps is especially similar to the E-GTSP with multiple depots. The challenge of the MTSPMT is the continuous movement of all targets. An arc distance is not only dependent on the time the arc is entered but also on the time the arc is left. Both these times are related to the positions of the incident targets.

Since solving time-dependent problems is very time-consuming, most research articles do not concentrate on exact procedures. In our research we confront different modeling approaches and examine their computation times on randomly generated test instances. Here, we concentrate on an offline approach for solving MTSPMT instances to global optimality. In real-world online

applications this can be used as a subroutine within a moving horizon approach.

3 Mathematical Models with Time

Some basic notation is introduced to formulate the MTSPMT as a mixed-integer optimization problem. We assume a finite time horizon $[0, T]$. The operating space is a square in the $\mathbb{R}^2$ with a side length $S \in \mathbb{R}_+$, but the following models may also be extended to the $\mathbb{R}^3$. Let $\mathcal{W} = \{1, \ldots, w\}$ be a set of salesmen. All salesmen start their tour at the same location o. Let $\mathcal{V} = \{1, \ldots, n\}$ be a set of nodes (targets, cities or customers), then $\mathcal{V}_o = \mathcal{V} \cup \{o\}$ and $\mathcal{A} \subseteq \mathcal{V}_o \times \mathcal{V}$ be a set of arcs (roads). The length of an arc depends on the time the arc is traversed and varies over time, since the nodes are moving. Thus, the distance for salesman k traveling from node i to node j starting at time s in i and arriving at time t in j is given by the function $c_{i,j,k} : [0, T] \times [0, T] \to \mathbb{R}_+ \cup \{\infty\}$. Since each target $i \in \mathcal{V}$ is assigned a visibility time window $[\underline{t}_i, \overline{t}_i]$, we have

$$c_{i,j,k}(s,t) = \infty \qquad \text{if and only if } \; s \notin [\underline{t}_i, \overline{t}_i] \text{ or } t \notin [\underline{t}_j, \overline{t}_j] \text{ or } \; s > t \text{ or } (t-s)\overline{v} < \left\| p^j(t) - p^i(s) \right\|_2, \tag{1}$$

where $p^i(s)$ and $p^j(t)$ are the respective locations of the targets at the times s and t and $\overline{v}$ is the maximum speed value of all salesmen. The arrival time of any salesman at a target is equal to his departure time at the same target, because waiting times are included in the traveling times. The depot o has the whole time horizon as visibility window. All arcs with a length less than infinity can be traveled with a speed less than or equal to $\overline{v}$. The goal is to reach each target from $\mathcal{V}$ by exactly one salesman such that the sum of all traveled distances of all salesmen is minimized.

3.1 A Time-Discrete Model

The time-discrete MILP formulation has already been addressed in [31]. For the convenience of the reader it is concisely presented in the sequel. The model consists of a multi-commodity flow formulation embedded in a time expanded network. For an explanation of time expanded networks see Ford and Fulkerson [10]. Here, we have discretized the whole time horizon in time steps. For each time step there is a time layer with a copy of each target that is visible in this time step. Let m be an integer number. The step size is defined by $\Delta := T/m$. Then the set of all time steps is $\mathcal{T} := \{0, \ldots, m\}$. With $\tilde{\mathcal{A}}$ we denote the set of arcs in the time-expanded network. An arc is only considered between different layers: the arc $(i, p, j, q) \in \tilde{\mathcal{A}}$, $i \in \mathcal{V}_o$, $j \in \mathcal{V}$, $p, q \in \mathcal{T}$ connects target i in layer p with target j in layer q. That means a salesman departs in i at time step p and arrives in j at time step q. For each salesman the arrival time at any node in $\mathcal{V}$ is equal to the departure time in the same node, since the waiting time is included in the traveling time, thus, salesmen do not necessarily use their maximum speed. Having discrete time steps $p, q \in \mathcal{T}$ we are able to evaluate the distance function c for arcs at these times

$$c^{p,q}_{i,j,k} := c_{i,j,k}(p\Delta, q\Delta).$$

We introduce a family of binary decision variables $x^{p,q}_{i,j,k} \in \{0, 1\}$. Here, $x^{p,q}_{i,j,k} = 1$ represents the decision of sending salesman k from i to j, departing at time step p in i and arriving in j at time step q. The objective function is to minimize the total traveled distances of all salesmen:

$$\sum_{k \in \mathcal{W}} \sum_{(i,p,j,q) \in \tilde{\mathcal{A}}} c^{p,q}_{i,j,k} \, x^{p,q}_{i,j,k} \quad \to \quad \min. \tag{2}$$

The demand constraint requires, that each node $j \in \mathcal{V}$ must be visited once by exactly one salesman:

$$\sum_{k \in \mathcal{W}} \sum_{(i,p,q):(i,p,j,q) \in \tilde{\mathcal{A}}} x^{p,q}_{i,j,k} = 1, \quad \forall \, j \in \mathcal{V}. \tag{3}$$

Each salesman $k \in \mathcal{W}$ can only start once from the depot:

$$\sum_{(j,p,q):(o,p,j,q) \in \tilde{\mathcal{A}}} x^{p,q}_{o,j,k} \leq 1, \quad \forall \, k \in \mathcal{W}. \tag{4}$$

The following flow constraints ensure the feasibility of time. Conservation is ensured at each node of the salesman tour except for the last one, this can be regarded as the sink of the flow:

$$\sum_{(i,p):(i,p,j,q)\in\tilde{\mathcal{A}}} x_{i,j,k}^{p,q} \geq \sum_{(i,p):(j,q,i,p)\in\tilde{\mathcal{A}}} x_{j,i,k}^{q,p}, \quad \forall\, j \in \mathcal{V}, q \in \mathcal{T}, k \in \mathcal{W}. \tag{5}$$

Summing up, we solve the following optimization problem:

$$\min\{(2) \mid (3), (4), (5), x \in \{0,1\}^{\tilde{\mathcal{A}}\times\mathcal{W}}\}. \tag{6}$$

The restrictions of the visibility time windows are embedded in the arcs, thus, arcs violating an involved time window constraint have infinite length. Usually TSP have to incorporate subtour elimination constraints. In fact, this point is included by the inherent time dependency. Since time evolves, there is no cycle in the underlying time expanded network and consequently subtour elimination constraints are not needed. Furthermore, the presented model (6) is not restricted to special shapes of the target trajectories. It can handle any trajectory, see for example Stieber and Fügenschuh [30]. Likewise, there is no need to restrict the speed function of the targets, the model is able to deal with varying target speeds. Apparently, an adverse effect of the above formulation is the increased problem size in case of a higher number of time steps (better accuracy), leading to a higher computational burden.

3.2 A Time-Continuous Model

The salesmen may intercept the targets at any point of their trajectories, thus, the time constraints have to be modelled by continuous variables. As opposed to the discrete model, the decision whether a salesman is able to reach the destination target of a direct link with its maximum speed has to be integrated into the model formulation. This is realized by applying the big-M method (similar to the Miller-Tucker-Zemlin [25] (MTZ) constraints for the TSP), containing continuous time variables and time restrictions. An obvious consequence of this approach is the fact, that we obtain an objective function value with best accuracy.

We introduce a family of binary decision variables $x_{i,j,k} \in \{0,1\}$, where $x_{i,j,k} = 1$ represents the decision of sending salesman k from i to j (independently of the time). Additionally, continuous time variables $t_{i,k} \in \mathbb{R}$ are defined to describe the arrival time of salesman k in node i. Here, the set of arcs is $\mathcal{A} \subseteq \mathcal{V}_o \times \mathcal{V}$. Now, we are able to formulate the continuous model.

The objective function is to minimize the total traveled distances of all salesmen:

$$\sum_{k\in\mathcal{W}} \sum_{(i,j)\in\mathcal{A}} c_{i,j,k}(t_{i,k}, t_{j,k})\, x_{i,j,k} \quad \to \quad \min\,. \tag{7}$$

Each node j must be visited once by exactly one salesman:

$$\sum_{k\in\mathcal{W}} \sum_{i:(i,j)\in\mathcal{A}} x_{i,j,k} = 1, \quad \forall\, j \in \mathcal{V}. \tag{8}$$

Each salesman k can only start once from the depot:

$$\sum_{j\in\mathcal{V}} x_{o,j,k} \leq 1, \quad \forall\, k \in \mathcal{W}. \tag{9}$$

Flow conservation is ensured by:

$$\sum_{i:(i,j)\in\mathcal{A}} x_{i,j,k} \geq \sum_{i:(j,i)\in\mathcal{A}} x_{j,i,k}, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}, \tag{10}$$

where again the $\geq$ only stands for the last node (sink). The following MTZ-type constraints guarantee time-feasibility, that means, if salesman k moves from i to j and arrives at $t_{i,k}$ in i, he cannot be earlier in j than $t_{i,k}$ plus the time he needs to travel from the position of i at $t_{i,k}$ to the position of j at $t_{j,k}$ using maximum speed $\overline{v}$. The time horizon T is the so called big-M constant in the big-M constraints

$$t_{i,k} + \frac{c_{i,j,k}(t_{i,k}, t_{j,k})}{\overline{v}} \leq t_{j,k} + T \cdot (1 - x_{i,j,k})\,, \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{11}$$

For the visibility time windows, the time variables have to satisfy the following bounds:

$$\underline{t}_j \leq t_{j,k} \leq \overline{t}_j, \quad \forall\, j \in \mathcal{V}_o, k \in \mathcal{W}. \tag{12}$$

Summarized, we aim to solve the following optimization problem:

$$\min\{(7) \mid (8), (9), (10), (11), (12), x \in \{0,1\}^{\mathcal{A}\times\mathcal{W}}, t \in \mathbb{R}^{\mathcal{V}_o\times\mathcal{W}}\}. \tag{13}$$

Basically, this model is the same as the time continuous model in [31]. The only changes made are some modeling refinements, e.g., departure and arrival time variables are replaced by arrival variables because possible waiting is included, constraints (9) are only considered for depot node o and bounds for the visibility time windows are added.

The presented time-continuous formulation of the MTSPMT is based on an arbitrary nonlinear continuous function $c_{i,j,k}$ for the distance between two distinct moving nodes. In order to apply a standard MILP solver such as CPLEX, we have to restrict the movement of the targets. To this end, we assume the trajectories to be straight lines and the speed of each target to be constant. To simplify matters, we use the same constant speed for all targets. Then $c_{i,j,k}$ represents the Euclidean distance between two points on two straight lines. With this, the above presented optimization problem (13) can be handled as a second order cone program (SOCP), see Boyd and Vandenberghe [4] for an overview. The constraints (11) define the cones and make the set of feasible solutions to be convex. In the sequel we present the adjusted SOCP formulation that we use for CPLEX.

We introduce the real auxiliary variables $c^x_{i,j,k}$ and $c^y_{i,j,k}$ for the x- and y-components of the Euclidean distance $c_{i,j,k}(t_{i,k}, t_{j,k})$; $a_{i,j,k}$ for the sum

$$a_{i,j,k} = \sum_{t_{i,k}\in[0,T]} \sum_{t_{j,k}\in[0,T]} c_{i,j,k}(t_{i,k}, t_{j,k})\, x_{i,j,k}$$

and finally $\overline{a}_{i,j,k}$ for the right hand side of the cone definition. Hence, we can formulate the following SOCP. The objective function is

$$\sum_{k\in\mathcal{W}} \sum_{(i,j)\in\mathcal{A}} a_{i,j,k} \quad \to \quad \min. \tag{14}$$

The constraints (8), (9), (10) and (12) remain unchanged. The constraints (11), that contain quadratic terms, are transformed into the following family of auxiliary conditions, where the trajectory of a target is represented by the convex combination of its start $(\underline{x}_i, \underline{y}_i)$ and end point $(\overline{x}_i, \overline{y}_i)$, see equations (15)–(18). We define $\Delta x_i = \overline{x}_i - \underline{x}_i$, $\Delta y_i = \overline{y}_i - \underline{y}_i$ and $\Delta t_i = \overline{t}_i - \underline{t}_i$.

$$c^x_{i,j,k} - \left(\left(\underline{x}_j + t_{j,k}\frac{\Delta x_j}{\Delta t_j} - \underline{t}_j\frac{\Delta x_j}{\Delta t_j}\right) - \left(\underline{x}_i + t_{i,k}\frac{\Delta x_i}{\Delta t_i} - \underline{t}_i\frac{\Delta x_i}{\Delta t_i}\right)\right) = \quad 0,$$
$$\forall\,(i,j) \in \mathcal{A}, i \neq o, k \in \quad \mathcal{W}. \tag{15}$$

$$c^x_{o,j,k} - \left(\left(\underline{x}_j + t_{j,k}\frac{\Delta x_j}{\Delta t_j} - \underline{t}_j\frac{\Delta x_j}{\Delta t_j}\right) - o^x\right) = 0, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}. \tag{16}$$

$$c^y_{i,j,k} - \left(\left(\underline{y}_j + t_{j,k}\frac{\Delta y_j}{\Delta t_j} - \underline{t}_j\frac{\Delta y_j}{\Delta t_j}\right) - \left(\underline{y}_i + t_{i,k}\frac{\Delta y_i}{\Delta t_i} - \underline{t}_i\frac{\Delta y_i}{\Delta t_i}\right)\right) = \quad 0,$$
$$\forall\,(i,j) \in \mathcal{A}, i \neq o, k \in \quad \mathcal{W}. \tag{17}$$

$$c^y_{o,j,k} - \left(\left(\underline{y}_j + t_{j,k}\frac{\Delta y_j}{\Delta t_j} - \underline{t}_j\frac{\Delta y_j}{\Delta t_j}\right) - o^y\right) = 0, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}. \tag{18}$$

The following constraints describe the condition of the uniform movement of the targets:

$$a_{i,j,k} \leq \overline{v}\,(t_{j,k} - t_{i,k} + T\,(1 - x_{i,j,k})), \quad \forall\,(i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{19}$$

The next conditions are needed to formulate the cone constraints:

$$\overline{a}_{i,j,k} = a_{i,j,k} + R\,(1 - x_{i,j,k}), \quad \forall\,(i,j) \in \mathcal{A}, k \in \mathcal{W}, \tag{20}$$

where R is equal to the diagonal of the considered square, $R = \lceil \sqrt{2}S \rceil$.
Finally, the cone constraints are given as:

$$(c^x_{i,j,k})^2 + (c^y_{i,j,k})^2 \leq (\overline{a}_{i,j,k})^2, \quad \forall\,(i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{21}$$

Summarized, the transformed SOCP reads the following:

$$\begin{aligned} \min\{(14) \quad | \quad & (8),(9),(10),(12),(15),(16),(17),(18),(19),(20),(21) \\ & x \in \{0,1\}^{\mathcal{A}\times\mathcal{W}}, t \in \mathbb{R}^{\mathcal{V}_o\times\mathcal{W}},\ c^x,\ c^y,\ a,\ \overline{a} \in \mathbb{R}^{\mathcal{A}\times\mathcal{W}}\}. \end{aligned} \tag{22}$$

Having this model formulation, a salesman is able to intercept a moving target at any possible point on its trajectory. In general, the objective function value of (22) is less than or equal to the objective function value of (6) for the same instance. However, the assumptions we had to made are severe, the model is only applicable to linear trajectories with constant speeds. In contrast to this, the time-discrete model can handle trajectories of any shape and speed. Moreover, model formulations based on big-M constraints usually have a weak linear programming relaxation and thus, more nodes have to be examined in the branch and bound tree, which slows down the solution process [5]. Additionally, due to the big-M constants numerical instabilities can occur in the solution procedure if the constants are not tight enough. In (22) there are two of such constants, see (19) and (20). Obviously, a comparison of both modeling approaches (discrete and continuous) is difficult, because of the different characteristics.

4 A Time Relaxation

For the next model formulation we concentrate on the time aspect in a different way by focussing on a time-free model. This means relaxing the time completely and later reintegrating parts of the time restrictions. The technique was first introduced by Fügenschuh et al. [11]. They successfully applied the method to the scheduling and routing of planes and tourist travel requests during fly-in safaris, which essentially is a VRPTW with pickup and delivery.

First of all, we perform a projection of (6) from the time-discrete variable space $\tilde{\mathcal{A}} \times \mathcal{W}$ to $\mathcal{A} \times \mathcal{W}$ (i.e., from $(\mathcal{V}_o \times \mathcal{T}) \times (\mathcal{V} \times \mathcal{T}) \times \mathcal{W}$ to $\mathcal{V}_0 \times \mathcal{V} \times \mathcal{W}$). The time-free counterpart of variables $x^{p,q}_{i,j,k}$ is simply $x_{i,j,k}$. While reducing all time-discrete arcs between two different targets to only one arc, we have to find a suitable length for this new arc to preserve the optimal solution. The minimum length of all time-discrete arcs between two targets serves as an appropriate length for the new arc. That means, for any salesman k and any two nodes i and j the distance coefficient $c_{i,j,k}$ is taken as

$$c_{i,j,k} = \min\{c^{p,q}_{i,j,k} \mid p,q \in \mathcal{T}\}. \tag{23}$$

With this, we have the following model in the time-free space:

$$\begin{aligned} \sum_{k\in\mathcal{W}} \sum_{(i,j)\in\mathcal{A}} c_{i,j,k}\, x_{i,j,k} \quad & \to \quad \min \\ \text{s.t.} \sum_{k\in\mathcal{W}} \sum_{i:(i,j)\in\mathcal{A}} x_{i,j,k} \quad & = \quad 1, \quad \forall\, j \in \mathcal{V} \\ \sum_{j:(o,j)\in\mathcal{A}} x_{o,j,k} \quad & \leq \quad 1, \quad \forall\, k \in \mathcal{W} \\ \sum_{i:(i,j)\in\mathcal{A}} x_{i,j,k} - \sum_{i:(j,i)\in\mathcal{A}} x_{j,i,k} \quad & \geq \quad 0, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W} \\ x \quad & \in \quad \{0,1\}^{\mathcal{A}\times\mathcal{W}}. \end{aligned} \tag{24}$$

This is a classical multi-commodity flow problem, which is easy to solve by standard MILP solvers. The optimal solution of the time-free model (24) is a lower bound to (6), because the distance

coefficients are computed by minimization. However, the reconstruction of a time-feasible solution from a time-free solution is not straightforward and not every time-free solution yields a time-feasible solution. For this purpose we have to examine every time-free solution, that we encounter during the solution process for time-feasibility. The examination is embedded in a branch-and-bound framework in order to prune nodes whose lower bounds exceed the current best solution value.

Given an optimal solution of (24), feasible times at which salesmen reach the targets have to be constructed from it. Assuming the objective function value of the constructed time-feasible solution is equal to the objective function value of the time-free solution, then the constructed solution is proven global optimal for (6). However, this rarely happens. It is likely, that a time-free solution is infeasible with respect to the time constraints. Thus, apart from an optimal time-free solution, we also have to investigate the other feasible time-free solutions in the branch-and-bound process. That means (24) serves as the master problem in the branch-and-bound framework. For any solution of the master problem, we try to construct a feasible counterpart with respect to the given time constraints. If the objective function value of this solution is better than previously found ones, it is stored. Then, the current time-free solution is treated as infeasible and cut off in the branch-and-bound process. Obviously, the generated cuts should take into account all possible salesmen permutations in order to prevent a repetition of the same time-free solution due to salesman symmetries. In case that there is no time-feasible counterpart for a time-free solution, we also have to cut off the time-free solution as well. In this way we are able to check all time free solutions. Additionally, the branch-and-bound tree can be pruned by exploiting lower bounds. This solution method is realized using the callback functionality of CPLEX. Analyzing the time-free solution before the construction phase also leads to an advantage in processing. For this we refer to the Section 5. For now, we concentrate on the construction of a time-feasible solution from a time-free solution.

Given a time-free solution, according to the variables set to one, we obtain a set of arcs for each salesman and call it a pretour. Note, a pretour may be disconnected and the pretour of some salesmen can be empty. A salesman is called active, if its pretour is not empty. The pretours of all active salesmen are extracted from the solution. A pretour is called feasible if it is a Hamiltonian path in the induced sub-graph starting at the depot o. For each feasible pretour a time-feasible tour is required.

4.1 A Time Relaxation with Discrete Time Feasibility Checking

Assuming we have a non-empty pretour. The construction of a time-feasible tour is done by setting up a checking sub-MILP. For the sub-MILP we consider all the time restrictions of the salesman, who belongs to the pretour. In more detail, we include only those time-dependent arcs, that have a time-free counterpart in the given pretour meaning the corresponding solution variable is nonzero. In the case that the checking sub-MILP for each active salesman results in a time-feasible tour an overall time-feasible tour is found. The best overall time-feasible tour solves (6).

The time-feasibility checking MILP is set up as a minimum-cost flow problem from a source to a sink for each active salesman separately. For an active salesman k its pretour defines the sequence of targets, k has to visit. Let us assume n_k is the number of targets k has to visit and $(v_1, v_2, \ldots, v_{n_k})$ is the sequence. Additionally depot position o is considered as the source of the flow and we extend the sequence by a node d, which serves as the sink. Thus, $\mathcal{V}^k = \{o = v_0, v_1, v_2, \ldots, v_{n_k}, v_{n_k+1} = d\}$ denotes the sequence of nodes considered for the minimum-cost flow problem of salesman k.

Since we have to consider only those time-expanded arcs, that correspond to an arc of the pretour, the checking MILP consists of all arcs, that go from depot position o to distinct positions of v_1 and from distinct positions of v_1 to distinct positions of v_2 and so on. Additionally, we have to introduce artificial time-discrete arcs from distinct positions of v_{n_k} to the sink d. That means, for each arc, that ends in v_{n_k} at time step p an arc is introduced from (v_{n_k}, p) to $(d, p+1)$. The distance of all arcs between v_{n_k} and d is zero. We denote this set of arcs for salesman k by $\mathcal{A}^k$. According to the time-discrete model, we introduce binary decision variables $x^{p,q}_{v_i,v_{i+1}}$ describing the decision of sending salesman k from target v_i to its successor v_{i+1} starting at time step p and arriving at time step q. Then, the time-feasibility checking MILP for an active salesman k is

formulated as follows:

$$\begin{aligned}
\sum_{(v_i,p,v_{i+1},q)\in\mathcal{A}^k} c^{p,q}_{v_i,v_{i+1}}\, x^{p,q}_{v_i,v_{i+1}} \quad &\to \quad \min \\
\text{s.t.} \sum_{(p,q):(o,p,v_1,q)\in\mathcal{A}^k} x^{p,q}_{o,v_1} \quad &= \quad 1 \\
\sum_{(p,q):(v_{n_k},p,d,q)\in\mathcal{A}^k} x^{p,q}_{v_{n_k},d} \quad &= \quad 1 \\
\sum_{p:(v_{j-1},p,v_j,q)\in\mathcal{A}^k} x^{p,q}_{v_{j-1},v_j} - \sum_{p:(v_j,q,v_{j+1},p)\in\mathcal{A}^k} x^{q,p}_{v_j,v_{j+1}} \quad &= \quad 0,\, \forall j \in \{1,\dots,n_k\}, q \in \mathcal{T} \\
x \quad &\in \quad \{0,1\}^{\mathcal{A}^k}.
\end{aligned} \tag{25}$$

The optimization problem (25) aims to find the shortest path from o to d. This kind of optimization problem can be solved in polynomial time by, e.g., Dijkstra's algorithm [7]. In case the checking MILP (25) results in a time-feasible tour for each active salesman, we are able to construct a total time-feasible solution for (6) by combining all salesman tours. If the checking MILP results in an infeasible solution for any of the active salesmen the construction process is aborted for the corresponding time-free solution.

4.2 A Time Relaxation with Continuous Time Feasibility Checking

According to our time-continuous model (13), there is also a time-relaxed variant with a continuous time-feasibility checking sub-MILP. For this variant we cannot use (24) directly, since its distance coefficients $c_{i,j,k}$ in the objective function are dependent on a discretization and on the time-discrete arcs, see (23). In a continuous model, these coefficients are not valid. Here, we have to replace the minimum time-expanded arclength by the real minimum length between any two trajectories. Thus, the distance coefficients $c_{i,j,k}$ are computed as follows: For an endpoint p_e^j of trajectory j we seek for the time interval $[t_1^i, t_2^i]$ for i, such that

$$t_1^i = \inf\{t \in [\underline{t}_i, \overline{t}_i] \mid k \text{ can travel from } p^i(t) \text{ to } p_e^j \text{ with speed } \overline{v}\}$$

and

$$t_2^i = \sup\{t \in [\underline{t}_i, \overline{t}_i] \mid k \text{ can travel from } p^i(t) \text{ to } p_e^j \text{ with speed } \overline{v}\},$$

where $p^i(t)$ is the position of i at time t. Let I be the union of the intervals for both endpoints of j and $J = [\underline{t}_j, \overline{t}_j]$. Then we compute the overall minimum distance between the trajectory of i reduced to the interval I and trajectory of j:

$$c_{i,j,k} = \min\{c_{i,j,k}(s,t) \mid s \in I,\ t \in J\}. \tag{26}$$

Then the continuous time-relaxed model is (24) with (26). Having this, we can formulate the feasibility checking sub-MILP for the continuous case. As for the continuous model (22) with the assumption of linear trajectories and constant target speed, the checking sub-MILP for an active salesman k can be modeled as a quadratic program. Here, we again use continuous time variables t_{v_i} to define the arrival time in v_i. With the target sequence $\{o = v_0, v_1, \dots, v_{n_k}\}$ and the corresponding set of arcs $\mathcal{A}^k$, we obtain the checking MILP for k as follows:

$$\begin{aligned}
\sum_{i=0}^{n_k-1} c_{v_i,v_{i+1}}(t_{v_i}, t_{v_{i+1}})\, x_{v_i,v_{i+1}} \quad &\to \quad \min \\
\text{s.t.} \quad c_{v_i,v_{i+1}}(t_{v_i}, t_{v_{i+1}}) \quad &\le \quad \overline{v}\,(t_{v_{i+1}} - t_{v_i}), \quad \forall\, i = 0,1,\dots,n_k-1, \\
t_{v_i} \quad &\in \quad [\underline{t}_i, \overline{t}_i], \quad \forall\, i = 1,\dots,n_k \\
x \quad &\in \quad \{0,1\}^{\mathcal{A}^k}.
\end{aligned} \tag{27}$$

Here, the function $c_{v_i,v_{i+1}}(t_{v_i}, t_{v_{i+1}})$ again denotes the Euclidean distance between the position of target v_i at time step t_{v_i} and the position of target v_{i+1} at time step $t_{v_{i+1}}$. In the objective function

all traveled distances are summed up, while in the restrictions time-feasibility is checked. That means, the travel speed, that k needs to traverse an arc with length $c_{v_i,v_{i+1}}(t_{v_i}, t_{v_{i+1}})$ in a time difference of $(t_{v_{i+1}} - t_{v_i})$, has to be at most $\overline{v}$, the maximum speed. In contrast to the proposed TC model (13), the time-free model with the continuous checking MILP (TFTC) does not contain any big-M constant.

5 Implementational Details

The solution method to solve the time-free problems (24) plus (23) and (24) plus (26) including the construction of feasible times for the salesman tours is explained in more detail. The model (24) (with either (23) or (26)) is called master problem and their solution procedure is embedded in a branch-and-bound framework. To check the solutions of the master problem and to produce the best time-feasible solution we use the callback utilities of CPLEX. We implement an instance of the BranchCallback and the LazyConstraintCallback. The latter one is a user-written callback, that can be applied to solve mixed-integer linear programs. Each time a candidate feasible solution of the master problem is found at a node in the branch-and-bound tree CPLEX' LazyConstraintCallback is invoked and constraints can be applied in a "lazy" fashion, i.e., only if they are violated.

In the LazyConstraintCallback we include a validation check of the candidate feasible solution of the master problem and the construction of the feasible times according to (25) and (27). Our callback algorithm is presented in Algorithm 1. A more detailed description of single steps is given in the sequel. In line 2-3 the objective function value of the master problem is checked. This value serves as a lower bound. If the lower bound at the current node is greater or equal to the best time-feasible objective function value found so far, we will quit the callback. Afterwards, the current node is pruned in the BranchCallback, which is subsequently invoked. The BranchCallback is only for pruning, branching decisions are left to CPLEX.

Since the master problem is a multi-commodity flow formulation, any solution of it presents a feasible flow but not necessarily a feasible tour, due to the lack of subtour elimination constraints. Therefore, in line 5-7 of Algorithm 1 each active pretour is checked against cycles. In case a cycle $\mathcal{C} \subset \mathcal{A}$ is found, it will be cut off by the following constraints:

$$\sum_{(i,j)\in\mathcal{C}} x_{i,j,k} \leq |\mathcal{C}| - 1, \quad \forall\, k \in \mathcal{W}. \tag{28}$$

There are other cut formulations as well. We tested another one[1], but it did not have any positive effect on the runtime.

Before setting up the checking MILP to create time-feasibility, we perform a validation ckeck for each pretour by exploiting the visibility time windows, see line 9-11. To test a pretour we start the interval propagation at its first node. In general, for any target v we check, if there are time steps in the visibility window of v, which can be used to arrive from its predecessor and to leave for its successor. All those time steps are collected by intersecting both the possible arrival interval and the possible departure interval of v. The intersected interval serves as the new visibility window for v and it is propagated to the succeeding node the same way. Since for the depot node o the whole time horizon is considered as visibility window, the whole visibility window for v_1 can be used as its *arrival interval*. Regarding the visibility window of v_2 we obtain a possible *departure interval* for v_1 with feasible time steps to depart for v_2. The next step is to generate a cut set from both intervals of v_1. This procedure is continued to the last node. The case, that the intersection interval at a certain node is empty, indicates that the pretour we started with is not time-feasible. An example of the interval propagation is visualized in Figure 1. Here, the *arrival interval* of node v_1 is the discrete interval $[2, 5]$. The *departure interval* to leave for v_2 with a speed of at most $\overline{v}$ is $[2, 3]$ with a possible arrival in v_2 in $[3, 4]$. A departure later than 3 in v_1 cannot reach v_2 within its visibility window of $[0, 4]$. An earlier departure is not possible due to the visibility window of v_1. Thus, the intersection of $[2, 5]$ and $[2, 3]$ leads to $[2, 3]$, which is considered as the new visibility window for v_1 when it comes to v_2. The procedure of time interval propagation has to be continued to the following targets, but it is important to use the updated time windows for predecessor nodes. In case there is an empty cut set at any target of the pretour, we have an infeasible interval. Thus, there is no time-feasible tour for the current pretour and we can reject the time-free solution. This is done by adding the following global cut to the master problem. For

[1]The number of arcs between the set of the cycle nodes and the remaining nodes were considered.

Algorithm 1: LazyConstraintCallback: Pretour validation check and time-feasibility construction.

Data: Time-relaxed x variables, best objective function value found so far *best_obj_val*
Result: If exists time-feasible tours for all salesmen

```
#Bounds exploitation
if current objective function value curr_obj_val ≥ best_obj_val then
    return;
#Cycle detection
if the pretour of an active salesman s contains a cycle then
    for all salesmen add a global cut;
    return;
#Validation check by interval propagation
if the pretour of an active salesman s is identified as time-infeasible then
    for all salesmen add a global cut;
    return;
#Time-Feasibility Check
initialize current solution curr_tour ← ∅; all_salesmen_feasible ← true;
initialize objective function value for time-feasible solution tour_val ← 0;
for each active salesman s ∈ 𝒲 do
    if all_salesmen_feasible ≠ true then
        break;
    if pretour of s is already in solution pool then
        get time-feasible tour r for s from solution pool;
        if r is infeasible then
            all_salesmen_feasible ← false;
    else
        set up MILP to compute time-feasible tour r for s;
        solve MILP;
        #r is a time-feasible tour of s;
        tour_val ← objective function value of MILP;
        curr_tour = curr_tour ∪ r;
        add pretour of s and time-feasible tour r to solution pool;
if all_salesmen_feasible = true then
    add global cut to prevent solution to be repeated by another salesmen permutation;
    if tour_val < best_obj_val then
        best_obj_val = tour_val;
        save curr_tour;
        return;
return;
```

an infeasible pretour $o = v_0$ to v_{n_s}, let $\mathcal{P} \subset \mathcal{A}$ be the corresponding sequence of arcs, then, we formulate the following constraint for each salesman to cut off this pretour:

$$\sum_{(i,j)\in\mathcal{P}} x_{i,j,k} \leq |\mathcal{P}| - 1, \quad \forall\, k \in \mathcal{W}. \tag{29}$$

Denote, that for any pair of anti-parallel arcs (i, j) and (j, i) we have

$$x_{i,j,k} + x_{j,i,k} \leq 1, \quad \forall\, k \in \mathcal{W}. \tag{30}$$

This means, for a variable in (29), which is bounded by 1, we can add the variable of the anti-parallel arc at no cost (30). Lifting $|P| - 1$ anti-parallel arcs to the cut (29), leads to:

$$\sum_{(i,j)\in\mathcal{P}} x_{i,j,k} + \sum_{(i,j)\in\mathcal{P}\setminus(v_{n_s-1},v_{n_s})} x_{j,i,k} \leq |\mathcal{P}| - 1, \quad \forall k \in \mathcal{W}. \tag{31}$$

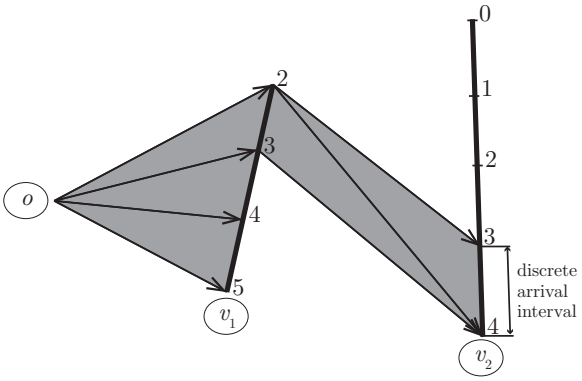

Figure 1: Interval propagation. This figure presents the depot position o and a given salesman tour consisting of the sequence o, v_1, v_2. Each target is visualized by its trajectory and the corresponding discrete time steps, which are given by numbers. The grey area describes the discrete *arrival* and *departure interval* between consecutive nodes.

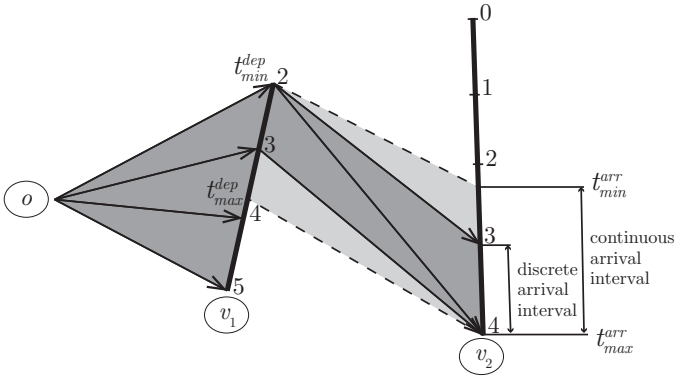

Figure 2: Interval propagation. This figure presents the depot position o and a given salesman tour consisting of the sequence o, v_1, v_2. Each target is visualized by its trajectory and the corresponding discrete time steps, which are given by black numbers. The grey area describes the discrete departure and arrival interval between consecutive nodes. The light grey area is the extended departure and arrival interval when considering continuous times.

Interval propagation for the time-feasibility checking with continuous times is done in a similar way. In general, the resulting arrival and departure intervals are slightly larger, see Figure 2. The travel time is not rounded to the next time step, its exact value is computed from the maximum salesman speed and the Euclidean distance between the corresponding positions of the targets. With this a salesman is able to arrive earlier and to depart later compared to the case of discrete time steps.

The computation of the new time window of a target again is an intersection of the arrival and the departure interval. The exact *arrival interval* depends on the previously updated time interval of the predecessor node and the departure interval depends on the visibility window of the successor. In Figure 2 the *arrival interval* of v_1 is the whole visibility window, since the predecessor node is the depot. For the *departure interval* of v_1 the values $t^{dep}_{min}, t^{dep}_{max}, t^{arr}_{min}$ and t^{arr}_{max} have to be identified. For v_1 we know that $t^{dep}_{min} = \underline{t}_{v_1}$ and $t^{arr}_{max} = \overline{t}_{v_2}$ in v_2, because waiting is permitted. Assuming a constant maximum speed we take the equation of uniform movement to compute t^{dep}_{max}, which is the latest possible departure in v_1:

$$\overline{v}\,(\overline{t}_{v_2} - t^{dep}_{max}) = \left\| p_{arr} - p_{dep} \right\|_2, \tag{32}$$

where p_{arr} and p_{dep} are the positions of the targets at the times $\overline{t}_{v_2}$ and t^{dep}_{max} respectively. Since the right hand side of the motion equation is depended on t^{dep}_{max} we have to square both sides and replace p_{arr} and p_{dep} by their trajectory parameterization. This leads to a quadratic equation, from which t^{dep}_{max} can be obtained. An intersection of both intervals of v_1, with are $[2, 5]$ and $[t^{dep}_{min}, t^{dep}_{max}]$) gives us the new time interval of feasible time steps.

The next step is to calculate the *arrival interval* $[t^{arr}_{min}, t^{arr}_{max}]$ of v_2 from the recently computed time interval of v_1. Here, the missing t^{arr}_{min} is again calculated by the equation of uniform movement, see (32). This interval is the arrival interval of v_2 when we move to the next nodes for interval propagation. The intervals $[t^{dep}_{min}, t^{dep}_{max}]$ and $[t^{arr}_{min}, t^{arr}_{max}]$ have to be computed accordingly for v_2 and v_3. This procedure is continued to the last node of the pretour or to the case an empty intersection interval has been detected. In the latter case there is no time-feasible salesman tour and the corresponding pretour is cut off for any salesmen permutation. In the feasible case the MILP (25) or (27) is set up to compute the corresponding times.

Each computed pretour and its counterpart with time are stored in a solution pool in order to prevent setting up and solving the same sub-MILP again and again for pretours occurring more than once. This is realized in Algorithm 1 in the lines 18-19 and 32. The existence of a solution pool forces us to use a single-threaded optimization instead of a multi-threaded one. The reason is, that in parallel mode it is not allowed to access data, which is not local. Another reason against parallel mode is, that CPLEX is not deterministic due to a different order of callback invocations for multiple runs of the same instance with the same parameter setting on the same platform. This would lead to an undesired, non-deterministic variability in the running times.

Finally, a time-feasible solution is found, when there is a time-feasible tour for each active salesman. This solution is returned to the master problem and the best objective function value found so far is saved. Then a global cut is added to the master problem in order to prevent a repetition of the current solution by other salesman permutations. Summing up, we have the time-discrete model (6) (TD), the time-continuous model (13) (TC), the time-free model (24) with time-discrete feasibility checking (25) according to Algorithm 1 (TFTD) and the time-free model (24) with time-continuous feasibility checking (27) according to Algorithm 1 (TFTC).

6 Computational Experiments

The presented models have either a discrete or a continuous handling of time. Often it depends on the application or on the computational burden which approach to choose. Both approaches have their specific characteristics, thus, it is not easy to do a comparison with each other. One thing is the handling of trajectories, here, the time-continuous models are restricted to the linear case to be solvable, while the time-discrete models are not. Another point is the difference in objective function values between discrete and continuous approaches. Despite of that we perform a runtime comparison of the proposed approaches. As a basis we concentrate on randomly generated linear target trajectories with constant speed for all our modeling variants.

6.1 Instance Generation

We use a set of randomly generated test instances. A test instance is specified by the number of salesmen, the number of moving targets and the distance of the time steps. Start and end points of the linear trajectories are generated randomly. The operating space is a square of size 500 length units and the trajectories are created with random lengths between 100 and 400 length units. Since it is very unlikely, that two hostile rockets meet in the air by chance, they would be deflected or destroyed by themselves before their destinations are reached. We do not support such a situation in the trajectory generation by prohibiting any pairwise intersections. Apart from this the targets are assigned a constant speed value of 32 length units per time unit, while the salesmen can travel at most 200 length units in one time unit. In [31] we observed, that instances are more difficult, if the difference between target speed and maximum salesmen speed is high. This is because the number of possible tours for the salesmen rises with an increased speed difference. Obviously, a power of 2 for the target speed is required to be able to create finer time-discretizations of the trajectories by introducing new time steps right in the middle of two existing ones. Following this, we create 3 different levels of time discretization. In particular, for the first discretization level called D32 a time step is introduced to the trajectories every 32 length units (target speed). Then, the same instances are generated with a two times finer discretization (D16). Here, the step size between two consecutive time steps is 16 length units and for the 4 times finer discretization (D8) time steps are included every 8 length units. Apparently, the size of the instances in terms of number of variables and constraints is increased with a higher number of time steps.

The current research deals with solvable test instances, that means no target will reach its upper time limit before being visited by a salesman. This is achieved by assigning the visibility time windows to the targets in such a way that one salesman is able to intercept all targets one after another. In all instances salesmen start their tours at the depot position o, an initial position located in the center of the operating space. In total, instances with a target number of 6, 8, 12, 16 and 20, with a salesmen number of 1, 2, 3, 4 and 6 and with discretization levels D32, D16 and D8 are created.

6.2 Computational Results

The proposed models and methods are not restricted to the two dimensional space. They can also be applied to the n-dimensional space, $n \in \mathbb{N}$. Furthermore our time-discrete models TD and TFTD are not restricted to linear trajectories, it is also possible to handle non-linear trajectories, see [30].

All proposed models are solved within the CPLEX framework. While the solution procedure of the time-free models TFTD and TFTC is customized by a LazyConstraintCallback and a BranchCallback of CPLEX, the instances modeled with TD and TC are solved without callbacks and directly by CPLEX' MILP and Barrier algorithms. The CPLEX parameters used for the optimization of the generated instances are listed in Table 1. For the time-free models, we turn off the node heuristic (HeurFreq) in order to save runtime, otherwise CPLEX would permanently check time-free solutions that are usually infeasible. Furthermore, we set the MIPEmphasis parameter to moving best bounds for the time-free models. For TD this setting would extremely slow down the computation, thus, to be fair we leave the MIPEmphasis parameter at its default value. Moreover, for the time-free master models we also turn off the cuts CPLEX creates, since our LazyConstraintCallback is producing cuts, when checking the pretours.

Since CPLEX' callbacks are not compatible with dynamic search, we turn it off for all branch-and-bound models. For several reasons we cannot use parallel optimization. It is not compatible with the solution pool, since it makes our callbacks non-deterministic. Another reason is, that CPLEX starts several callbacks concurrently and even if the solution is already found, optimization terminates after finishing all callbacks and their synchronization. Furthermore, CPLEX cannot guarantee the same order of callback invocation for multiple runs, in this sense parallel optimization may lead to different runtimes. We use sequential optimization mode instead. A time limit of one hour is enforced to our experiments. All other CPLEX parameters than listed in Table 1 were used with their default values.

Table 1: CPLEX parameter settings.

model	CPLEX parameter	parameter value
TD, TC,	EpGap	0.0

model	CPLEX parameter	parameter value
TFTD and TFTC	WorkMem	12288.0
master	Param::Threads	1
	Param::TimeLimit	3600
TFTD and TFTC	HeurFreq	-1
master	MIPEmphasis	3
	CutsFactor	1.0
TFTD subMILP	EpGap	0.0
TFTC subMILP	EpGap	0.0

The computational experiments were carried out on an Apple Mac Pro computer running the MacOS 10.12.6 operating system with an Intel Xeon E5 running at 3.5 GHz on 6 cores, 12 MB L3 cache, and 128 GB 1066 MHz DDR3 RAM. The version of CPLEX we used was 12.7.1 [6]. Our aim is to evaluate the presented models according to their computation times for solving the generated instances. As the runtime of an instance we consider the time, CPLEX requires to compute the global proven optimal solution, including the time for the callbacks, see Algorithm 1. In order to compare runtimes with a time limit, we compute a comparable score *sc*. It takes into account the runtime, which is at most 3600 seconds and the remaining gap, which is at most 1. It is computed as $sc = \frac{runtime}{3600} + gap$. The score takes values between 0 and 2. In case the score is less than 1, the optimization has finished within an hour and the gap is 0. In case the score is more than 1, the optimization has aborted with a gap equal to $(sc - 1) \cdot 100$ percent.

To summarize the computed results we aggregate the score values of each instance setting (number of targets 'nbt', number of salesmen 'nbs', discretization level 'dl' and number of target copies 'nbtc'), by using the arithmetic mean of all the 5 instances. The values are given in Table 2. The first 4 columns in Table 2 specify the instance setting. Columns 5 to 8 contain the arithmetic means of the computed scores of the proposed models TD, TFTD, TC and TFTC.

Table 2: Arithmetic mean of the score values for instances with 6, 8, 12, 16 and 20 targets. Values are rounded.

instance				arithmetic mean of the score			
nbt	nbs	dl	nbtc	TD	TFTD	TC	TFTC
6	1	D32	40	0.0000	0.0000	|	|
6	1	D16	75	0.0000	0.0000	0.0001	0.0000
6	1	D8	144	0.0004	0.0000	|	|
6	2	D32	40	0.0000	0.0000	|	|
6	2	D16	75	0.0000	0.0000	0.0002	0.0000
6	2	D8	144	0.0001	0.0000	|	|
6	3	D32	40	0.0000	0.0000	|	|
6	3	D16	75	0.0000	0.0000	0.0004	0.0000
6	3	D8	144	0.0001	0.0000	|	|
6	4	D32	40	0.0000	0.0000	|	|
6	4	D16	75	0.0000	0.0000	0.0009	0.0001
6	4	D8	144	0.0002	0.0000	|	|
6	6	D32	40	0.0000	0.0001	|	|
6	6	D16	75	0.0001	0.0002	0.0047	0.0003
6	6	D8	144	0.0003	0.0002	|	|
8	1	D32	56	0.0000	0.0000	|	|
8	1	D16	103	0.0003	0.0001	0.0002	0.0002
8	1	D8	198	0.0018	0.0001	|	|
8	2	D32	56	0.0000	0.0001	|	|
8	2	D16	103	0.0001	0.0001	0.0011	0.0002
8	2	D8	198	0.0008	0.0001	|	|
8	3	D32	56	0.0000	0.0001	|	|
8	3	D16	103	0.0001	0.0001	0.0039	0.0003
8	3	D8	198	0.0005	0.0001	|	|
8	4	D32	56	0.0000	0.0008	|	|
8	4	D16	103	0.0001	0.0008	0.0128	0.0022
8	4	D8	198	0.0006	0.0010	|	|

instance				arithmetic mean of the score			
nbt	nbs	dl	nbtc	TD	TFTD	TC	TFTC
8	6	D32	56	0.0000	0.0629	|	|
8	6	D16	103	0.0001	0.0654	0.1047	0.1527
8	6	D8	198	0.0010	0.0696	|	|
12	1	D32	81	0.0001	0.0015	|	|
12	1	D16	150	0.0009	0.0024	0.0020	0.0085
12	1	D8	287	0.0678	0.0050	|	|
12	2	D32	81	0.0001	0.0069	|	|
12	2	D16	150	0.0013	0.0109	0.0574	0.0415
12	2	D8	287	0.0302	0.0162	|	|
12	3	D32	81	0.0001	0.0691	|	|
12	3	D16	150	0.0004	0.2119	0.3819	0.5683
12	3	D8	287	0.0152	0.2233	|	|
12	4	D32	81	0.0001	0.7151	|	|
12	4	D16	150	0.0004	0.7143	0.9848	0.8521
12	4	D8	287	0.0491	0.7186	|	|
12	6	D32	81	0.0001	1.1880	|	|
12	6	D16	150	0.0021	1.2040	1.5360	1.2460
12	6	D8	287	0.1922	1.2100	|	|
16	1	D32	105	0.0001	0.1208	|	|
16	1	D16	194	0.0043	0.1719	0.0275	0.3429
16	1	D8	371	0.4482	0.2437	|	|
16	2	D32	105	0.0001	0.3293	|	|
16	2	D16	194	0.0037	0.4433	0.7179	0.8809
16	2	D8	371	0.2364	0.5085	|	|
16	3	D32	105	0.0003	0.8943	|	|
16	3	D16	194	0.0035	0.9181	1.5640	0.9821
16	3	D8	371	0.1601	0.9196	|	|
16	4	D32	105	0.0003	1.1680	|	|
16	4	D16	194	0.0012	1.1860	1.6700	1.23409
16	4	D8	371	0.0263	1.1880	|	|
16	6	D32	105	0.0002	1.2740	|	|
16	6	D16	194	0.0027	1.2840	1.7840	1.3260
16	6	D8	371	0.2084	1.2720	|	|
20	1	D32	129	0.0002	0.6203	|	|
20	1	D16	237	0.0073	0.7238	0.1745	0.9849
20	1	D8	454	0.4504	0.7713	|	|
20	2	D32	129	0.0003	0.9712	|	|
20	2	D16	237	0.0066	0.9601	1.4840	1.1720
20	2	D8	454	0.2665	0.9796	|	|
20	3	D32	129	0.0005	1.0054	|	|
20	3	D16	237	0.0206	1.0836	1.7520	1.1960
20	3	D8	454	0.2541	1.1169	|	|
20	4	D32	129	0.0004	1.2080	|	|
20	4	D16	237	0.0130	1.2140	1.8540	1.2400
20	4	D8	454	0.3227	1.2140	|	|
20	6	D32	129	0.0008	1.2860	|	|
20	6	D16	237	0.1339	1.2800	1.8960	1.3140
20	6	D8	454	0.3655	1.2940	|	|

The results indicate that small instances can be solved quickly by all four proposed model variants and the proposed solution methods, while the running times for bigger instances are modest for the time-free and continuous variants. Furthermore, the averaged score values show, that the time-free models TFTD and TFTC and the continuous model TC are very sensitive to the number of salesmen. If the salesmen number grows the runtimes of the models will also increase. However, the TD model is very sensitve to the discretization level.

The score values of the most interesting model pairs TD-TC, TD-TFTD, TC-TFTC and TFTD-TFTC are visualized as scatter plots, see Figure 3. Here, we used all score values of each single instance. In the case of comparing a time-discrete with a time-continuous model we used the same time-continuous score values for the discretization level D32, D16 and D8.

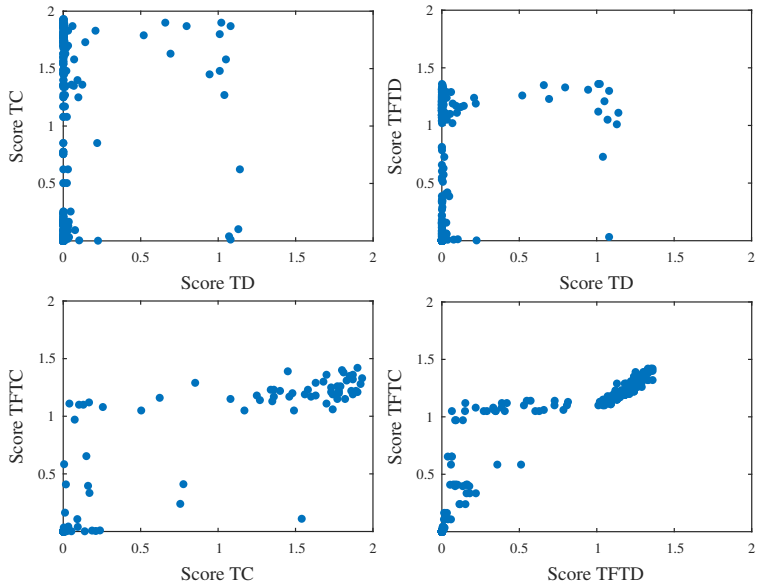

Figure 3: Visualization of the score values of each generated instance between different model pairs.

The results suggest that the continuous problems are more difficult to solve than the discrete ones, that leads us to a separate comparison of discrete and continuous variants. Comparing TD and TFTD, the time free variant can outperform TD only for instances with a small target number or a small salesman number and for the finest discretization, but there is also a D8 instance with 16 targets and 2 salesmen where TFTD is faster than TD. Usually, the scores of TFTD and TFTC have a wide range of values, that means several instances can be solved quickly but some instances need an enormous amount of time. In the continuous case the time free variant is better than TC for big instances with multiple salesmen. Here the dual bounds for TFTC are usually better than for TC, which also gives a better gap and thus a better score. The gap value can be obtained from the score by taking $gap = sc - 1$.

7 Conclusion

We addressed the MTSPMT, a dynamic variant of the TSP, where multiple salesmen are searching for their tours in a system with continuously moving targets. We considered different model formulations, which can be separated by discrete and continuous time behavior on one hand and on the other hand by time and time-free approaches. For the time-free variants we presented an exact branch-and-cut algorithm. Due to the different ways of modeling the variants have different characteristics inherent. The time-discrete model TD is sensitive to the discretization precision but nevertheless very fast in computation, especially for large instances and even for a high discretization. However, for small instances TFTD performed better when considering a high discretization. The best performance is obtained with one salesman. The runtimes of the models TC, TFTC and TFTD correlate with the number of salesmen and therefor resulting in high score values. Comparing both continuous variants TFTC has a better evaluation for large instances due to better dual bounds. Despite TD performed best for large discrete instances of the MTSPMT, there are large instances where TFTD is better. In future research we want to have a closer look on those instances. So far we only dealt with solvable instances, our next step will be to investigate non-solvable ones, that are instances, where it is not possible to reach all targets within their visibility time windows.

References

[1] Abeledo, H.G., Fukasawa, R., Pessoa, A.A., Uchoa, E.: The time dependent traveling salesman problem: polyhedra and algorithm. Mathematical Programming Computation **5**, 27–55 (2013)

[2] Ahrens, B.: The tour construction framework for the dynamic travelling salesman problem. In: SoutheastCon 2015, pp. 1–8 (2015). DOI 10.1109/SECON.2015.7132999

[3] Albiach, J., Sanchis, J.M., Soler, D.: An asymmetric tsp with time windows and with time-dependent travel times and costs: An exact solution through a graph transformation. European Journal of Operational Research **189**(3), 789–802 (2008)

[4] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)

[5] Codato, G., Fischetti, M.: Combinatorial benders cuts. In: D. Bienstock, G. Nemhauser (eds.) Integer Programming and Combinatorial Optimization, *Lecture Notes in Computer Science*, vol. 3064, pp. 178–195. Springer Berlin Heidelberg (2004). DOI 10.1007/978-3-540-25960-2_14

[6] IBM ILOG CPLEX 12.7.1 Documentation available at `https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html` (09/2017)

[7] Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1**(1), 269–271 (1959). DOI 10.1007/bf01386390

[8] Englot, B.J., Sahai, T., Cohen, I.: Efficient tracking and pursuit of moving targets by heuristic solution of the traveling salesman problem. In: CDC, pp. 3433–3438. IEEE (2013)

[9] Fleischmann, B., Gietz, M., Gnutzmann, S.: Time-varying travel times in vehicle routing. Transportation Science **38**(2), 160–173 (2004)

[10] Ford, L.R., Fulkerson, D.R.: Constructing Maximal Dynamic Flows from Static Flows. Operations Research **6**(3), 419–433 (1958)

[11] Fügenschuh, A., Nemhauser, G., Zeng, Y.: Scheduling and routing of fly-in safari planes using a flow-over-flow model. In: M. Jünger, G. Reinelt (eds.) Facets of Combinatorial Optimization, pp. 419–447. Springer Berlin Heidelberg (2013)

[12] Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York (1979)

[13] Ghiani, G., Improta, G.: An efficient transformation of the generalized vehicle routing problem. European Journal of Operational Research **122**, 11–17 (2000)

[14] Haghani, A., Jung, S.: A dynamic vehicle routing problem with time-dependent travel times. Computers & Operations Research **32**(11), 2959–2986 (2005)

[15] Helvig, C., Robins, G., Zelikovsky, A.: Moving-target tsp and related problems. In: A.P. G. Bilardi G. F. Italiano, G. Pucci (eds.) Proceedings of the European Symposium on Algorithms, *Lecture Notes in Computer Science*, vol. 1461, pp. 453–464. Springer Verlag, Berlin (1998)

[16] Helvig, C., Robins, G., Zelikovsky, A.: The moving-target traveling salesman problem. Journal of Algorithms **49**(1), 153–174 (2003)

[17] Ichoua, S., Gendreau, M., Potvin, J.Y.: Vehicle dispatching with time-dependent travel times. European Journal of Operational Research **144**(2), 379–396 (2003)

[18] Jiang, Q., Sarker, R., Abbass, H.: Tracking moving targets and the non-stationary traveling salesman problem. Complexity International **11**, 171–179 (2005)

[19] Jindal, P., Kumar, A., Kumar, S.: Multiple target intercepting traveling salesman problem. International Journal on Computer Science and Technology **2**(2), 327–331 (2011)

[20] Jung, S., Haghani, A.: Genetic algorithm for the time-dependent vehicle routing problem. Transportation Research Record: Journal of the Transportation Research Board **1771**, 164–171 (2001)

[21] Laporte, G., Mercure, H., Nobert, Y.: Generalized travelling salesman problem through n sets of nodes: the asymmetrical case. Discrete Applied Mathematics **18**(2), 185–197 (1987)

[22] Lawler, E., Lenstra, J., Rinnooy, A., Shmoys, D.: The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization. John Wiley and Sons, Chichester, New York (1985)

[23] Malandraki, C., Daskin, M.S.: Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. Transportation Science **26**(3), 185–200 (1992)

[24] Mancini, S.: Time dependent travel speed vehicle routing and scheduling on a real road network: The case of torino. Transportation Research Procedia **3**, 433–441 (2014)

[25] Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer programming formulation of traveling salesman problems. Journal of the ACM (JACM) **7**(4), 326–329 (1960)

[26] Noon, C.E., Bean, J.C.: An efficient transformation of the generalized traveling salesman problem. INFOR: Information Systems and Operational Research **31**(1), 39–44 (1993)

[27] Picard, J.C., Queyranne, M.: The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. Operations Research **26**(1), 86–110 (1978)

[28] Reinelt, G.: The Traveling Salesman: Computational Solutions for TSP Applications. Springer Verlag, Berlin (1994)

[29] Soler, D., Albiach, J., Martínez, E.: A way to optimally solve a time-dependent vehicle routing problem with time windows. Operations Research Letters **37**(1), 37–42 (2009)

[30] Stieber, A., Fügenschuh, A.: The multiple traveling salesmen problem with moving targets and nonlinear trajectories. In: N. Kliewer, J.F. Ehmke, R. Borndörfer (eds.) Operations Research Proceedings 2017, pp. 489–494. Springer International Publishing, Cham (2018)

[31] Stieber, A., Fügenschuh, A., Epp, M., Knapp, M., Rothe, H.: The multiple traveling salesmen problem with moving targets. Optimization Letters **9**(8), 1569–1583 (2014)

[32] Sundar, K., Rathinam, S.: Generalized multiple depot traveling salesmen problem polyhedral study and exact algorithm. Computers & Operations Research **70**, 39–55 (2016)

[33] Woensel, T.V., Kerbache, L., Peremans, H., Vandaele, N.: A queueing framework for routing problems with time-dependent travel times. Journal of Mathematical Modelling and Algorithms **6**(1), 151–173 (2007)

IMPRESSUM

Brandenburgische Technische Universität Cottbus-Senftenberg
Fakultät 1 | MINT - Mathematik, Informatik, Physik, Elektro- und Informationstechnik
Institut für Mathematik
Platz der Deutschen Einheit 1
D-03046 Cottbus

Professur für Ingenieurmathematik und Numerik der Optimierung
Professor Dr. rer. nat. Armin Fügenschuh

E fuegenschuh@b-tu.de
T +49 (0)355 69 3127
F +49 (0)355 69 2307

Cottbus Mathematical Preprints (COMP), ISSN (Print) 2627-4019
Cottbus Mathematical Preprints (COMP), ISSN (Online) 2627-6100

www.b-tu.de/cottbus-mathematical-preprints
cottbus-mathematical-preprints@b-tu.de
doi.org/10.26127/btuopen-4824